вания. В данной работе предложен математический аппарат, с помощью которого делается такая формализация.

Вергазов Р.И.

ТЕСТОВОЕ ЗАДАНИЕ ДЛЯ ПРОВЕРКИ ЗНАНИЙ ПО АЛГОРИТМИЧЕСКИМ ЯЗЫКАМ

ellekta@yahoo.com

Пензенский Государственный Университет

г. Пенза

При создании тестов для дисциплин тесно связанных с использованием алгоритмических языков возникает проблема автоматизации проверки умений и навыков тестируемых в области создания программ. В качестве примера можно привести следующее задание:

“Реализовать на языке C++ функцию, меняющую местами значения своих аргументов. Использовать следующий прототип функции: void changeIt(int& val1, int& val2).”

Попытки автоматизировать проверку знаний с помощью существующих типов тестовых заданий (ТЗ) не увенчались успехом так, как не существует типа ТЗ способного автоматически определить правильность выполнения задания. Первой проблемой, встающей при автоматизации проверки, является наличие нескольких вариантов решения одной и той же задачи. Ответом на приведенный выше пример ТЗ могут являться следующие варианты реализации заданной функции:

1) Используя дополнительную переменную.

```
void changeIt(int& val1, int&val2)  {
        int tmp = val1;
        val1 = val2;
        val2 = tmp;
    }
```

2) Используя операцию XOR.

```
void changeIt(int& val1, int&val2) {
        val1 = val2 ^ val1;
        val2 = val2 ^ val1;
        val1 = val2 ^ val1;
    }
```

Оба варианта будут удовлетворять требованиям задания. При увеличении сложности задания количество правильных вариантов реализации будет возрастать, и их учет будет представлять очень сложную задачу. Второй проблемой является то, что тестируемый может называть имена вводимых им переменных как угодно. Таким образом, проверка полученного решения с помощью заранее заданного одного или нескольких шаблонов, используя ТЗ краткого ответа, не может применяться при оценке данного задания.

Для автоматизации проверки предлагается ввести и реализовать в компьютерной системе тестирования (КСТ) новый тип ТЗ – ТЗ на компиляцию ответа. ТЗ на компиляцию ответа представляет собой ТЗ открытой формы, в котором тестируемому предлагается на заданном языке программирования реализовать какой-либо алгоритм. После формирования тестируемым ответа происходит его компиляция и выполнение КСТ тес-

тов для проверки правильности функционирования и реализации созданной программы. Тесты для проверки правильности функционирования программы предлагается встраивать на этапе подготовки ТЗ. Таким образом, этап подготовки ТЗ на компиляцию включает не только создание формулировки задания, но и написание функции позволяющей протестировать работу созданной программы. Приведем код такой функции для описанного выше примера:

```
bool checkIt() {
        int var1 = 0; int var2 = 1;
        changeIt(var1,var2);
        bool flag = false;
        if (var1 == 1 && var2 == 0) flag = true;
        return flag;
}
```

Таким образом, окончательный текст программы поступающей на компиляцию будет таковым:

```
#include <iostream>
using std::cout;
void changeIt(int& val1, int&val2)  {…}
bool checkIt() {…}
int main(int argc, char* argv[])  {
        bool res = checkIt();
        if (res) cout <<“Passed”;
                else cout << “Failed”;
        return 0;
}
```

При успешной компиляции созданной программы и ее запуске в консоль будет выведено сообщение “Passed” – в случае успешного прохождения теста или “Failed” – в случае неправильно сформулированного ответа.

При наличии в программе синтаксических ошибок информация о них предоставляется тестируемому для последующего исправления и повторной попытки компиляции ответа. В случае наличия логических ошибок ТЗ, которые выявляются тестами, задание оценивается как невыполненное. Отдельно необходимо рассмотреть случай, когда логические ошибки могут привести к “зависанию” выполнения тестов программы. Например, реализуем умышленное зависание функции changeIt(…) вставкой бесконечного цикла:

```
void changeIt(int& val1, int&val2)  {
        for (;;);
}
```

В таком случае в реализации КСТ необходимо предусмотреть средства контроля за выполнением тестов созданной программы на правильность, чтобы предотвратить действия, приводящие к этому явлению. В случае если длительность процесса тестирования полученной программы на правильность превышает определенную величину, можно сделать вывод о возникновении логической ошибки выражающейся в бесконечном цикле или рекурсии и аварийно завершить данный процесс.

Другим аспектом создания ТЗ на компиляцию является ограничение тестируемого на использование стандартных классов или функций языка, на котором составляется программа. Для этого при подготовке ТЗ необходимо указать ключевые слова, которые не должны встречаться в теле про-

граммы создаваемой тестируемым и четко формировать требования к получаемому решению.

В целом, использование ТЗ на компиляцию позволяет облегчить труд преподавателя, переложив оценку заданий ориентированных на проверку умений и навыков на КСТ.

Выжанов Э.А.

ПЕРСПЕКТИВНЫЕ ТЕХНОЛОГИИ ОБУЧЕНИЯ - ОСНОВА ПРОГРЕССА

УГТУ-УПИ

г. Екатеринбург

Новейшие технологии обучения оперируются на научную, учебную информацию, которые должны быть:

1. долговременными (архивы знаний);
2. качественными (содержательными, полезными);
3. достаточными (объёмными) и наглядными;
4. доступными;

Текущие стандарты образования, базовые образования нацелены на традиционные формы обучения. Сегодня мы напичканы шаблонами стереотипов, прописных истин, безусловных канонов и стандартов. Кто успел их нам внушить? Что жизнь не требует мыслительных усилий...? А где активная роль личности? Сегодня мы напичканы шаблонами стереотипов, прописных истин, безусловных канонов и стандартов. Кто успел их нам внушить? Что жизнь не требует мыслительных усилий...?

Качество и количество «средств обучения» должно резко измениться. Педагог уже не «диктатор», а творческий исполнитель педагогических услуг, а «заказчиком» выступает сам обучающийся. С точки зрения само-